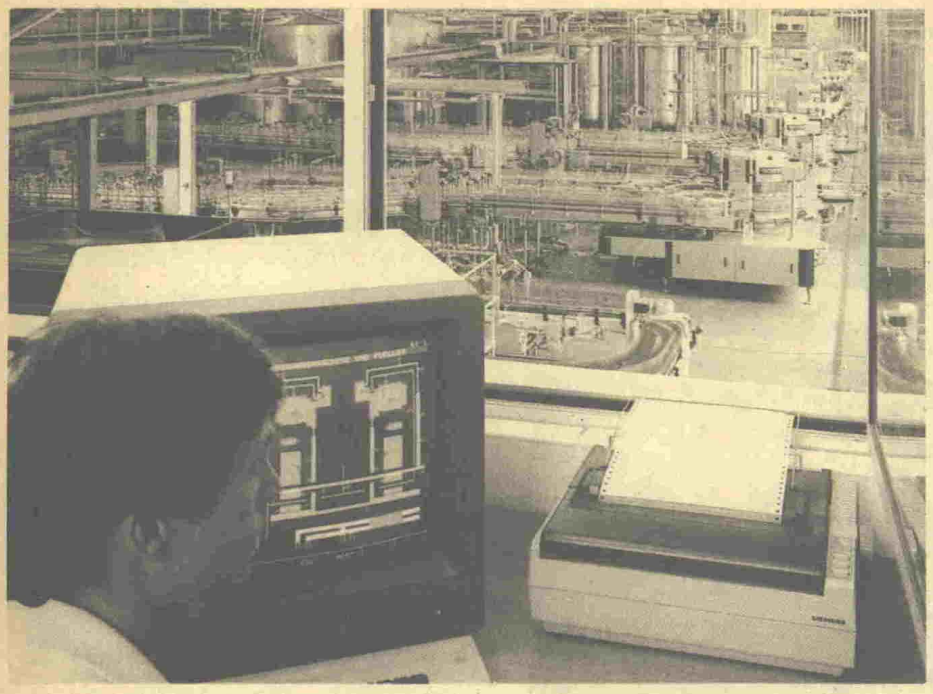
(12

การพัฒนาระบบซอฟต์แวร์ขนาดใหญ่

ครรชิต มาลัยวงศ์

เพื่อนชาวเยอรมันคนหนึ่งของผมโทรศัพท์มาปรึกษาหารือว่า ต้องการนำคอมพิวเตอร์ไปใช้งาน อยากทราบว่าในเมืองไทยนั้นมีคอมพิวเตอร์ของเยอรมันขายหรือเปล่า ผมตอบว่าผมไม่ทราบรายละเอียด แต่เห็นมีขายอยู่ยี่ห้อหนึ่งคือ นิกซดอร์ฟ เป็นเครื่องขนาดเล็กและใช้ในงานธุรกิจมากมายหลายแห่ง โดยเฉพาะที่มีการประชาสัมพันธ์มากก็คือระบบสำหรับสร้างระบบสารสนเทศแบบไทย ๆ ซึ่งน่าสนใจไม่น้อย เพื่อนของผมถามต่อไปว่า แล้วเครื่องขนาดใหญ่ของซีเมนส์ล่ะ มีขายบ้างไหม ผมเกาศีรษะแกรกเพราะแม้จะรู้จักชื่อเสียงของซีเมนส์มาตั้งแต่เล็ก แต่ก็ไม่เคยได้ทราบเลยว่า ซีเมนส์ผลิตคอมพิวเตอร์ขนาดใหญ่หรือมีใครใช้เครื่องของซีเมนส์ในไทย ผมลังเลอยู่หนึ่งอึดใจ แล้วก็ตอบไปว่าไม่มี



ผมรู้สึกอึดอัดใจเพราะไม่ทราบว่าตัวเองตอบเพื่อนไปถูกต้องหรือเปล่า เพราะทางเขาก็ต้องการรู้เร็ว ๆ ส่วนทางผมก็ไม่มีเวลาจะค้นหาคำตอบให้ แต่คงเป็นด้วยโชคเคราะห์ที่มักจะมาช่วยผมได้ทันเวลา เพราะยังไม่ทันข้ามวัน ผมก็ได้รับหนังสือเชิญจากท่านอาจารย์ประพัฒน์ อุทโยภาส ขุนพลคอมพิวเตอร์แห่งสถาบันวิจัยวิทยาศาสตร์และเทคโนโลยีแห่งประเทศไทย ให้ผมไปร่วมฟังซิมโปเซียมหรือการบรรยายวิชาการเกี่ยวกับฮาร์ดแวร์และซอฟต์แวร์ ซึ่งทางสถาบันฯ ร่วมจัดกับบริษัทซีเมนส์ ที่ห้องกำแพงเพชร โรงแรมเซ็นทรัลพลาซา ในวันเสาร์ที่ 28 พฤษภาคม แน่นอนครับลองมีโอกาสเช่นนี้ผมย่อมไม่ปล่อยไปเฉย ๆ อย่างน้อยก็จะได้เก็บอะไร ๆ ที่น่ารู้น่าสนใจมาเล่าให้ท่านผู้อ่านทราบไปด้วย

เดิมทีหัวข้อคำบรรยายนั้นกำหนดไว้สามหัวข้อใหญ่คือ การพัฒนาระบบซอฟต์แวร์ขนาดใหญ่ การจัดการโครงการพัฒนาระบบขนาดใหญ่ และระบบ CAD/CAM แต่เมื่อถึงเวลาจริงทางเยอรมันก็ขอตัดเรื่องการจัดการโครงการออกไป ด้วยความเสียดายของผู้ฟังหลายคน การบรรยายวิชาการคราวนี้นั้นฟรีครับ แต่น่าเสียใจที่มีผู้รู้ข่าวและมาฟังน้อยไปหน่อย ถ้ามากันมาก ๆ ก็คงจะถ่ายทอดความคิดที่น่าสนใจไปใช้ได้อีกมาก อย่างน้อยก็จะได้รู้กันมากขึ้นว่าเขาสร้างซอฟต์แวร์ใหญ่ ๆ กันอย่างไรในโลกแห่งความเป็นจริง

ผมจะไม่นำรายละเอียดของการบรรยายมาเล่าละครับ แต่จะขอเชิญไปร่วมสนทนากับ ดร.เวอร์เนอร์ แซมเมอร์ ผู้ซึ่งเป็นผู้อำนวยการอาวุโสในด้านคอมพิวเตอร์ของฝ่ายวิจัยและพัฒนาแทน ทั้งนี้ก็เพื่อจะได้รับฟังความคิดเห็นที่น่าสนใจเกี่ยวกับการพัฒนางานคอมพิวเตอร์ในปัจจุบัน ดร.แซมเมอร์ผู้นี้เดิมทีกำหนดว่าจะมาบรรยายเรื่องการบริหารโครงการที่ถูกตัดทิ้งไปนั่นและครับ

"ดร.แซมเมอร์ครับ ผมทราบมาว่าทางซีเมนส์ยังไม่ได้มีตัวแทนขายคอมพิวเตอร์ หรือเข้ามาจับธุรกิจด้านนี้ในเมืองไทยเลย ไม่ทราบว่าจริงไหมครับ" ผมถาม

"จริงครับ ในเอเชียนี่เรามีตัวแทนและสาขาในฮ่องกง อินเดีย ญี่ปุ่น เกาหลี สิงคโปร์ และไต้หวัน ส่วนประเทศไทยนั้นเรายังไม่มีตัวแทน" ผู้อำนวยการอาวุโสในวัยกลางคนราวห้าสิบตอบด้วยภาษาอังกฤษสำเนียงเยอรมัน

"แล้วทางบริษัทสนใจจะเข้ามาทำธุรกิจในเมืองไทยบ้างหรือเปล่าครับ"

"ก็สนใจสิครับ มิฉะนั้นเราคงไม่ยกทีมเข้ามาบรรยายวิชาการร่วมกับสถาบันวิจัยฯ แน่ ๆ" ดร.แซมเมอร์ตอบยิ้ม ๆ

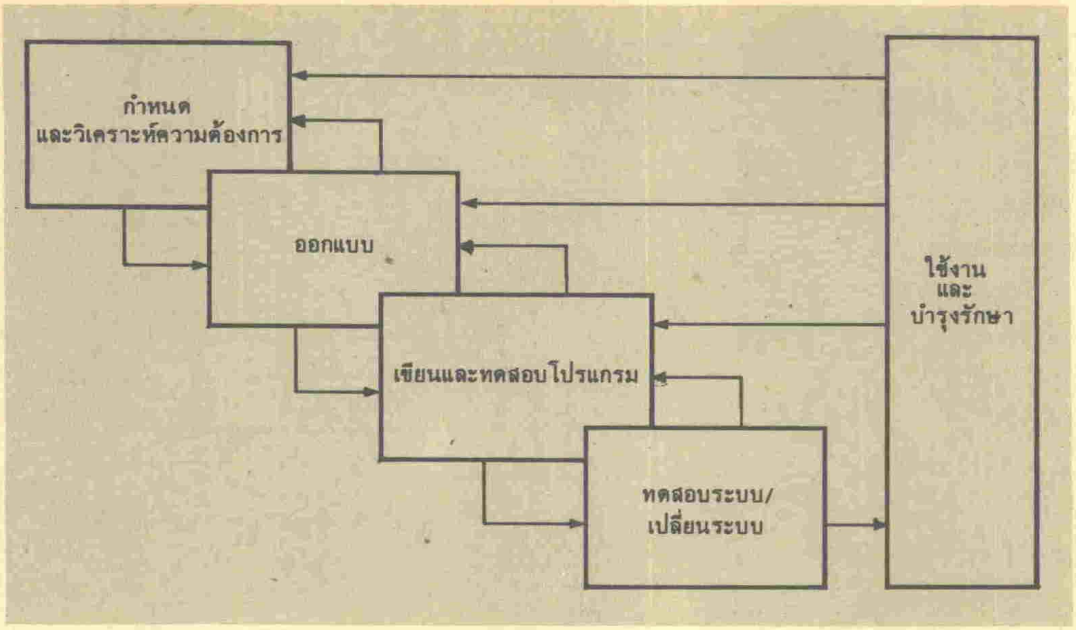
"ทางซีเมนส์ตอนนี้มีผลิตภัณฑ์อะไรบ้างครับที่อยากจะส่งเสริม"

"เราก็มีเมนเฟรมที่ใช้ระบบปฏิบัติการของเราเอง แล้วก็พวกสถานีงานสำหรับใช้ในงานวิศวกรรม ซึ่งเรามีหลายโมเดลเช่น รุ่น WS 30-200/220 และรุ่น WS 30-570/580 รุ่น 200 และ 220 นั้นเป็นรุ่นเล็กหน่อยใช้ไมโครโปรเซสเซอร์ MC 68020 มีหน่วยความจำตั้งแต่ 2 ถึง 8 เมกะไบต์ มี virtual address space 64 เมกะไบต์ ส่วนรุ่น 570/580 นั้นก็ใช้ MC 68020 เหมือนกันแต่มี virtual address space ถึง 2 กิกะไบต์" ดร.แซมเมอร์อธิบาย ซึ่งฟังดูแล้วก็รู้สึกสมศักดิ์ศรีที่จะเป็นสถานีงานได้จริง ๆ

"งานบรรยายวันนี้ส่วนใหญ่เป็นเรื่องของระบบซอฟต์แวร์ขนาดใหญ่ ไม่ทราบว่าทางซีเมนส์สร้างระบบอะไรครับที่พอจะเรียกว่าเป็นระบบใหญ่ได้"

"ก็ระบบ computer-aided design ที่จะใช้กับพวกสถานีงานนี่แหละครับ" ดร.แซมเมอร์ตอบ "ระบบ CAD ของเราใช้คำสั่งถึง 3 ล้านคำสั่ง และใช้ทีมงานถึง 300 คน อย่างนี้จะเรียกว่าใหญ่ได้ไหมครับ"

ผมหัวเราะแล้วลองคิดคำนวณในใจ ถ้านำโปรแกรมที่มีความยาวขนาดสามล้านคำสั่งมาพิมพ์บนกระดาษต่อเนื่องบรรทัดละคำสั่ง 60 บรรทัดต่อหน้า ก็จะต้องใช้กระดาษถึง 50,000 แผ่น หรือเท่ากับกระดาษทั้งหมด 25 กล่อง ใครก็ตามที่เห็นว่าโปรแกรมขนาดนี้เป็นโปร–



รูปที่ 1 โมเดลน้ำตกของวัฏจักรระบบงาน

แกรมเล็กก็คงจะไม่ใช่คนธรรมดาแล้ว คงเป็นยอดอภิมหาซูเปอร์โปรแกรมเมอร์แน่ ๆ ไม่ต้องดูอะไรอื่น โปรแกรมที่หนาเพียงไม่กี่ร้อยหน้า เรายังต้องประสบปัญหามากมายในการจัดทำให้มันสำเร็จถูกต้อง นี่ยาวขนาด 25 กล่องปัญหาจะเพิ่มพูนขึ้นกี่สิบเท่าก็ลองคิดดู

"คุณใช้วิธีอะไรในการพัฒนาโปรแกรมใหญ่ ๆ เหล่านี้ครับ" ผมถามขึ้นหลังจากใจลอยคิดถึงขนาดของโปรแกรมไปอึดใจหนึ่ง

"เราก็ใช้เทคนิควัฏจักรซอฟต์แวร์นี่แหละครับ" ดร.แซมเมอร์ตอบ "คือเริ่มจากการศึกษาความต้องการหรือ Requirement Analysis พยายามดูว่าผู้ใช้ระบบต้องการอะไร แล้วก็เขียนเป็นเอกสารขึ้น สำหรับงานในขั้นต่อไปคืองานออกแบบหรือ Design ต่อจากนั้นก็เป็นงาน Implementation หรือการทำแบบซึ่งมีลักษณะนามธรรมนั้นให้กลายเป็นโปรแกรมที่มีลักษณะเป็นรูปธรรม พูดง่าย ๆ ก็คืองานเขียนโปรแกรมและทดสอบแต่ละโปรแกรมนั่นเอง เมื่อเสร็จแล้วก็มาถึงขั้นการทดสอบระบบโดยนำระบบทั้งหมดมาทดสอบร่วมกัน แล้วก็ถึงขั้นสุดท้ายคือการใช้และแก้ไขปรับปรุงระบบ ถ้าหากเกิดความผิดพลาดหรือมีความต้องการเพิ่มเติม"

วัฏจักรระบบงานนั้นเป็นหลักการที่ใช้กันมาอย่างแพร่หลายและถือว่าเป็นพื้นฐานสำคัญอย่างหนึ่งของการพัฒนาระบบและซอฟต์แวร์ทั้งหลาย การรู้วัฏจักรระบบงานทำให้เราคาดคะเนตัวงานและปริมาณงานต่าง ๆ ได้ถูกต้อง เปรียบเสมือนตัวเรานี่แหละครับ พวกเราทุกคนย่อมรู้ว่าเมื่อเกิดมาแล้วก็ต้องเป็นทารกที่ยังปกป้องเลี้ยงตัวเองไม่ได้ ต่อมาจึงเป็นเด็กที่จะต้องศึกษาเล่าเรียนในระดับชั้นต่าง ๆ จนกว่าจะจบหลักสูตรถึงขั้นที่สามารถประกอบอาชีพได้ จึงหางานทำ ถ้าโชคไม่ร้ายเกินไปหางานทำได้จึงทำงานเก็บหอมรอมริบสร้างเนื้อสร้างตัว แต่งงานมีลูกเต้าแล้วก็ตาย การขาดความรู้ของวัฏจักรชีวิตที่ถูกต้องก็จะทำให้เกิดปัญหาแปลก ๆ เช่น เฒ่าทารก เด็กแก่แดด ชิงสุกก่อนห่าม อะไรพวกนี้แหละครับ ซึ่งไม่เป็นผลดีต่อวัฏจักรชีวิตเลย

ข้อที่ควรสังเกตอีกอย่างหนึ่งก็คือวัฏจักรระบบงานนั้นโดยสาระแล้วเหมือนกัน แต่โดยรายละเอียดแล้วขึ้นอยู่กับว่าใครจะย่อยวัฏจักรออกมาเป็นกี่ขั้นกี่ตอน ซึ่งอาจแตกต่างกันได้ สำหรับกรณีของทางซีเมนส์นี้ อาจเขียนเป็นแผนภาพไดอะแกรมให้เข้าใจได้ง่ายดังรูปที่ 1 ตามรูปนี้จะเห็นว่างานแต่ละขั้นนั้นอาจเหลื่อมล้ำกันได้ เนื่องจากเราไม่สามารถกำหนดได้ชัดเจนว่า นี่คืองานกำหนดความต้องการซึ่งถ้าทำเสร็จแล้วก็แล้วกันไป ระหว่างกำหนดความต้องการเราอาจนึกภาพการออกแบบไปด้วย หรือระหว่างการออกแบบเราอาจย้อนกลับมาหาความต้องการเพิ่มเติมก็ได้ ส่วนลูกศรที่ย้อนกลับนั้นแสดงการหวนกลับไปทำในขั้นตอนเดิมอีก จะเป็นด้วยงานที่กำลังทำอยู่มีอุปสรรค ผิดพลาด ไม่สมบูรณ์ จนต้องกลับไปทบทวนงานในขั้นตอนเดิมอย่างใดก็ได้ โมเดลแบบนี้เราเรียกว่าเป็นโมเดลน้ำตกครับ

"แล้วทางซีเมนส์ใช้วิธีการทำต้นแบบหรือ Prototyping บ้างหรือเปล่าครับ" ผมถามต่อ

"ใช้ครับ เราก็ใช้ในงานบางงานที่เหมาะสม" ดร.แซมเมอร์ตอบสั้น ๆ

"กรุณาขยายความสักนิดสิครับ" ผมต้องคะยั้นคะยอให้ได้ความมาก ๆ

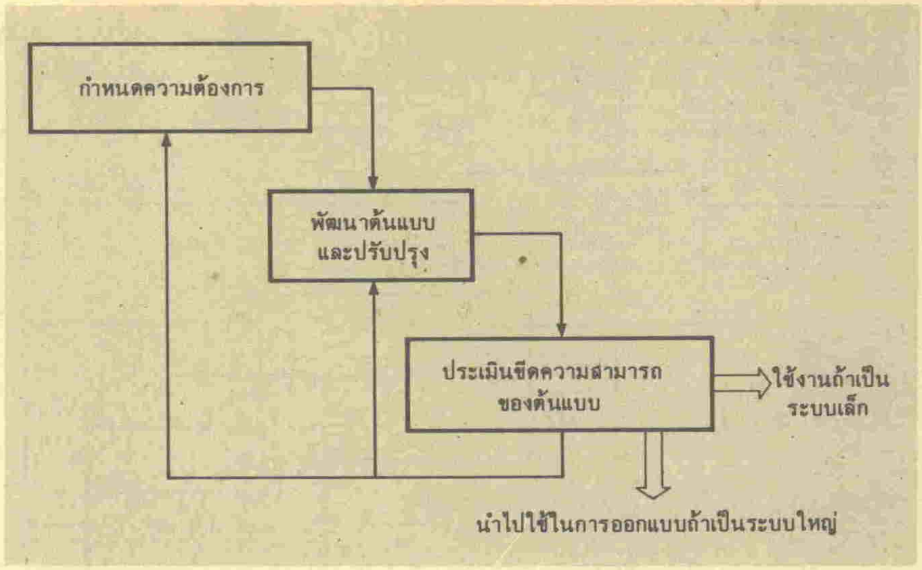
"คือการทำต้นแบบนั้น เราใช้ในช่วงของการกำหนดความต้องการครับ" ดร.แซมเมอร์อธิบาย "เป็นการกำหนด

functional requirement อย่างคร่าว ๆ ของระบบในรูปแบบที่ผู้ใช้จะเข้าใจได้ง่าย เช่น ในรูปของฟอร์แมตที่ปรากฏทางจอภาพ ผู้ใช้ต้องการเห็นอะไรเราก็สร้างจอภาพนั้นให้ดู มีเมนูต่าง ๆ เสร็จสรรพ พอกดปุ่มเลือกเมนูแล้วก็จะเห็นจอภาพตามที่ต้องการ เรียกว่าเป็นการทำต้นแบบขึ้นใช้แสดงคุณสมบัติของระบบที่ต้องการจะสร้าง ถ้าผู้ใช้ไม่ชอบหรือเห็นว่าไม่ถูกต้องกับความต้องการ เราก็เขียนหรือออกแบบจอภาพใหม่ แล้วนำไปให้ผู้ใช้ดูใหม่ จนกว่าจะพอใจ"

"ถ้าผู้ใช้พอใจก็เป็นระบบใช้การได้เลยหรืออย่างไรครับ"

"ยังครับ เราต้องทำความเข้าใจเรื่องนี้ก่อนว่า การทำต้นแบบนั้นเราต้องใช้เครื่องมือพัฒนาต้นแบบซึ่งเป็นภาษารุ่นที่สี่ หรือไม่ก็เป็นพวก Application Generator ซึ่งเครื่องมือพวกนี้แม้จะใช้สร้างระบบต้นแบบได้ แต่มันมี overhead สูง ดังนั้นโปรแกรมหรือระบบที่ได้จึงมักจะมีประสิทธิภาพต่ำ ทำงานช้า ถ้าหากระบบของคุณเล็ก และไม่ได้ใช้งานประจำแล้วละก็ คุณก็คงไม่แคร์เรื่องประสิทธิภาพ มันทำงานช้าหน่อยก็ไม่เป็นไร อย่างนี้คุณก็เอาต้นแบบไปตบแต่งอีกนิดหน่อยแล้วก็ใช้ได้เลย แต่ถ้าหากเป็นระบบใหญ่ ๆ เราจำเป็นจะต้องทำให้ประสิทธิภาพสูง ๆ อย่างนี้เราก็เอาต้นแบบมาดูในฐานะที่เป็น Requirement เท่านั้น เราจะต้องออกแบบแล้วเข้าสู่ขั้นตอนของวัฏจักรระบบงานเหมือนเดิม" ดร.แซมเมอร์ตอบ พร้อมกับเอื้อมมือไปหยิบแฟ้มแผ่นใสมาพลิกหารูปการทำต้นแบบให้ผมดู ดังที่ผมจำมาวาดเป็นรูปที่ 2

"ผมพอจะเข้าใจละครับ" ผมตอบ "คราวนี้ผมขอเปลี่ยนมาถามเรื่องของการควบคุมโครงการบ้าง งานใหญ่ ๆ อย่างนี้คุณใช้วิธีอะไรในการคาดคะเนหรือประมาณกำลังคนและกำลังงานล่ะครับ"



รูปที่ 2 ขั้นตอนการทำต้นแบบ

"เราใช้เทคนิค Cocomo ของเบมครับ คุณคงเคยได้ยินชื่อเทคนิคนี้มาบ้างแล้ว" ดร.แซมเมอร์ตอบ

"ครับ แต่โคโคโมเป็นวิธีการที่อาศัยสถิติประสบการณ์การทำโครงการซอฟต์แวร์ของบริษัทในอเมริกานะครับ คุณจะเอาตัวเลขมาใช้ได้หรือ" ผมซักไซร้ต่อ

"เราใช้สถิติของเราเองสิครับ" ดร.แซมเมอร์ตอบยิ้ม ๆ "พอเราทำโครงการต่าง ๆ เสร็จแล้ว เราก็เก็บสถิติเอาไว้ตลอดเวลาว่า โครงการขนาดนั้น ๆ ใช้ทีมงานกี่คน ทำงานกี่วันกี่เดือนเสร็จ"

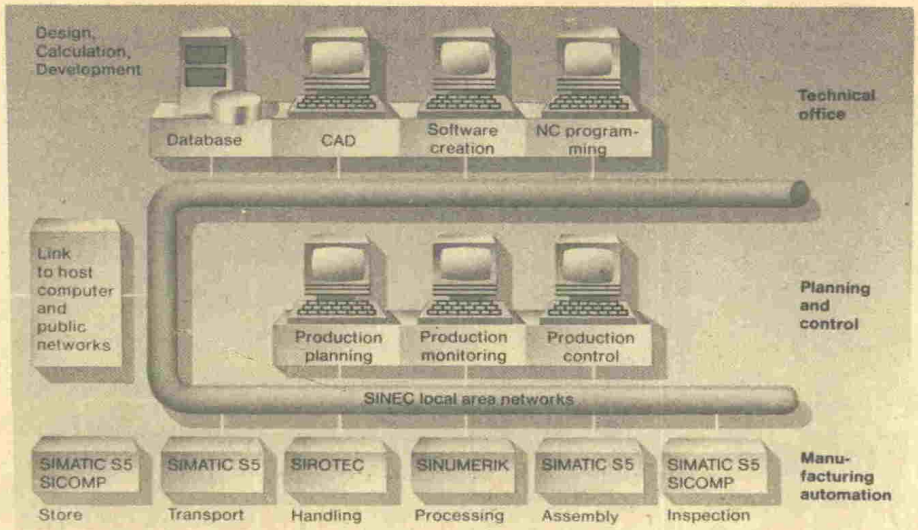
ถึงตรงนี้ผมคงจะต้องขยายความเรื่องของโคโคโมให้ท่านผู้อ่านได้ทราบแนวคิดเพิ่มสักหน่อย คำว่า COCOMO นี้เป็นคำย่อของ COnstructive COst MOdel ซึ่งเป็นแบบจำลองสำหรับคาดคะเนปริมาณงานของโครงการพัฒนาซอฟต์แวร์ รายละเอียดของโคโคโมนี้ปรากฏในหนังสือ Software Engineering Economics ของนายบาร์รี ดับลิว. เบม ตีพิมพ์เมื่อปี 1981 แบบจำลองโคโคโมนี้มีอยู่ด้วยกันสามระดับคือ

แบบจำลองที่ 1 เป็นแบบจำลองพื้นฐานสำหรับคาดคะเนแรงงานและค่าใช้จ่ายในการพัฒนาซอฟต์แวร์โดยคำนวณจากขนาดของโปรแกรม มีหน่วยนับเป็นจำนวนคำสั่ง (Line of Code หรือ Delivered Source Instruction)

แบบจำลองที่ 2 เป็นแบบจำลองระดับกลางซึ่งคำนวณแรงงานและค่าใช้จ่ายในการพัฒนาซอฟต์แวร์จากขนาดของโปรแกรม ร่วมกับตัวแปรอีกชุดหนึ่งที่จะทำให้เกิดค่าใช้จ่ายในโครงการเช่น ลักษณะของผลผลิตซอฟต์แวร์ ฮาร์ดแวร์ กำลังคน และลักษณะของโครงการ

แบบจำลองที่ 3 เป็นแบบจำลองระดับสูงที่ขยายออกจากแบบที่ 2 และรวมการประเมินผลกระทบของตัวแปรในระดับที่ 2 เข้ากับงานแต่ละขั้นของโครงการ เช่น งานออกแบบ งานเขียนโปรแกรม เป็นต้น

นอกจากจะมีสามระดับนี้แล้ว เบมยังแบ่งซอฟต์แวร์ออกเป็นสามกลุ่มตามลักษณะความยากง่ายของซอฟต์แวร์ กลุ่มแรกคือกลุ่ม Organic เป็นซอฟต์แวร์ง่าย ๆ ใช้ทีมงานขนาดเล็กทำก็ได้และความต้องการของผู้ใช้ก็ไม่ซับซ้อนหรือ



ยุ่งยากมากเท่าใดนัก กลุ่มที่สองคือกลุ่ม Semi-detached เป็นงานที่ยากขึ้น ใหญ่มากขึ้น ต้องใช้ทีมงานที่มีประสบการณ์มากขึ้น อีกทั้งข้อกำหนดความต้องการก็มีทั้งที่ตายตัวและยืดหยุ่นได้บ้าง เช่น งาน On-line และงานที่ใช้ฐานข้อมูลเป็นต้น ส่วนกลุ่มที่สามคือกลุ่ม embedded เป็นงานที่มีฮาร์ดแวร์ ซอฟต์แวร์ และการปฏิบัติงานที่ค่อนข้างตายตัวมาก ๆ เช่นงานพัฒนาซอฟต์แวร์สำหรับควบคุมการจราจรทางอากาศ เป็นต้น

โมเดลพื้นฐานของโคโคโมใช้สมการดังต่อไปนี้

$$E = a(KDSI)\ exp(b)$$

$$D = c(E)\ exp(d)$$

โดยที่ E เป็นกำลังงานทั้งหมดที่จะต้องใช้ในการพัฒนาซอฟต์แวร์คิดเป็น man-month

D เป็นเวลาทั้งหมดที่ต้องใช้ในการพัฒนาซอฟต์แวร์แต่ไม่นับช่วงวิเคราะห์ความต้องการ หน่วยเป็นเดือน

KDSI เป็นจำนวนคำสั่งทั้งหมด หน่วยเป็นพันคำสั่ง

a,b,c,d เป็นพารามิเตอร์ดังแสดงในตารางที่ 1

เมื่อเรามีโครงการซอฟต์แวร์โครงการหนึ่งซึ่งเราอยากรู้ว่าควรจะใช้คนกี่คนและทำกี่วันเสร็จ เราก็เพียงแต่กำหนดลักษณะโครงการว่าง่ายหรือยาก คาดคะเนขนาดซอฟต์แวร์เป็นจำนวนคำสั่ง แล้วใช้สูตรข้างต้นหา E และ D ออกมา ครับนี่ก็คือสาระของโคโคโมอย่างสั้น ๆ คราวนี้ขอเชิญติดตามคำสนทนาของผมกับ ดร.แซมเมอร์ต่อไปครับ

"การใช้โคโคโมนั้นเราต้องรู้จำนวนคำสั่งหรือ Delivered Source Instruction ที่จะเขียน จึงจะกำหนดจำนวน man-month หรือ effort ได้ถูกต้อง ปัญหามีอยู่ว่าก่อนที่เราจะลงมือพัฒนาระบบหรือสร้างซอฟต์แวร์นั้น เรายังไม่ทราบเลยว่าโปรแกรมมันจะยาวแค่ไหน เรื่องนี้คุณมีความเห็นอย่างไรบ้างครับ" ผมถามขึ้น เพราะเรื่องนี้เป็นจุดอ่อนสำคัญในแนวคิดของการใช้ความยาวของโปรแกรมเป็นตัวกำหนด man-month

"เรื่องนี้อยู่ที่การคิดฟังก์ชันของระบบในตอนแรกนะครับ" ดร.แซมเมอร์ตอบ "เราจะต้องกำหนดฟังก์ชันให้แน่ชัดที่สุด และจะต้องย่อยฟังก์ชันลงไปให้ละเอียดพอควร ซึ่งนี่ก็ต้องอาศัยประสบการณ์มากทีเดียว ถ้าคุณไม่มีประสบการณ์คุณก็จะกำหนดไม่ได้ว่าโปรแกรมควรจะยาวสักเท่าใด"

"แต่โคโคโมนี้ใช้กับวิธีทำต้นแบบไม่ได้ใช่ไหมครับ"

"ใช่ครับ ใช้ไม่ได้แน่ ๆ" ดร.แซมเมอร์ตอบ "ตอนที่เบมคิดเทคนิคโคโคโมออกมานั้นยังไม่มีใครใช้วิธีทำต้นแบบนี่ครับ" เขาทิ้งท้ายอย่างขำ ๆ

"การพัฒนาซอฟต์แวร์ขนาดใหญ่ ๆ นั้นคงจะต้องระวังด้านการทดสอบมากพอสมควร ไม่ทราบว่าคุณมีเทคนิคหรือข้อคิดอะไรบ้างครับเกี่ยวกับการทดสอบนี้" ผมถามต่อ

"ก็ไม่มีอะไรมากครับ" ดร.แซมเมอร์ตอบ "ก่อนอื่นเราต้องเข้าใจปรัชญาการทดสอบเสียก่อนคือ เราต้องทดสอบว่าโปรแกรมทำงานถูกต้องจริง ๆ ไม่ใช่ตรวจแต่เพียงว่าไม่มีที่ผิดในโปรแกรม เพราะโปรแกรมหนึ่งนั้นถึงจะไม่มีที่ผิดแต่ก็อาจจะทำงานไม่ถูกต้อง แต่ถ้าโปรแกรมถูกต้องแล้วโปรแกรมนั้นย่อมจะไม่ผิด"

"ในการทดสอบนั้น เราจำเป็นจะต้องแยกคนทดสอบหรือเปล่าครับ คือเขียนโปรแกรมคนหนึ่งแล้วทดสอบอีกคนหนึ่ง"

"มันแล้วแต่การทดสอบครับ" ดร.แซมเมอร์ตอบ "ถ้าเป็นการเขียนและทดสอบแต่ละโปรแกรมเราก็ใช้คน ๆ เดียวกันไม่ต้องแยก แต่พอถึงการทดสอบระบบเราจำเป็นจะต้องมีคนทดสอบแยกต่างหาก"

"การเตรียมการทดสอบนั้นควร

ประเภทโครงการ	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

ตารางที่ 1 ค่าพารามิเตอร์สำหรับแบบจำลอง COCOMO

จะเริ่มเมื่อใดดีครับ" ผมถามอีก

"ถ้าเป็นการทดสอบระบบ เราควรเตรียมข้อมูลทดสอบได้ในช่วงของการออกแบบ เพราะในช่วงนี้เราทราบข้อกำหนดของระบบดีแล้ว แต่ถ้าเป็นการทดสอบโปรแกรมเราก็จะไปกำหนดรายละเอียดของการทดสอบในช่วงของการออกแบบและเขียนโปรแกรม"

"เวลาเราทดสอบโปรแกรมกับระบบนี่เราต้องเก็บผลลัพธ์เอาไว้อ้างอิงไหมครับ"

"เก็บซีครับ ถ้าเป็นการทดสอบระบบซึ่งเรากำหนดข้อมูลต่าง ๆ เอาไว้อย่างสมบูรณ์นั้น เราต้องเก็บผลลัพธ์ของการทดสอบเอาไว้ด้วย ไม่ใช่ทดสอบแล้วก็ทิ้ง"

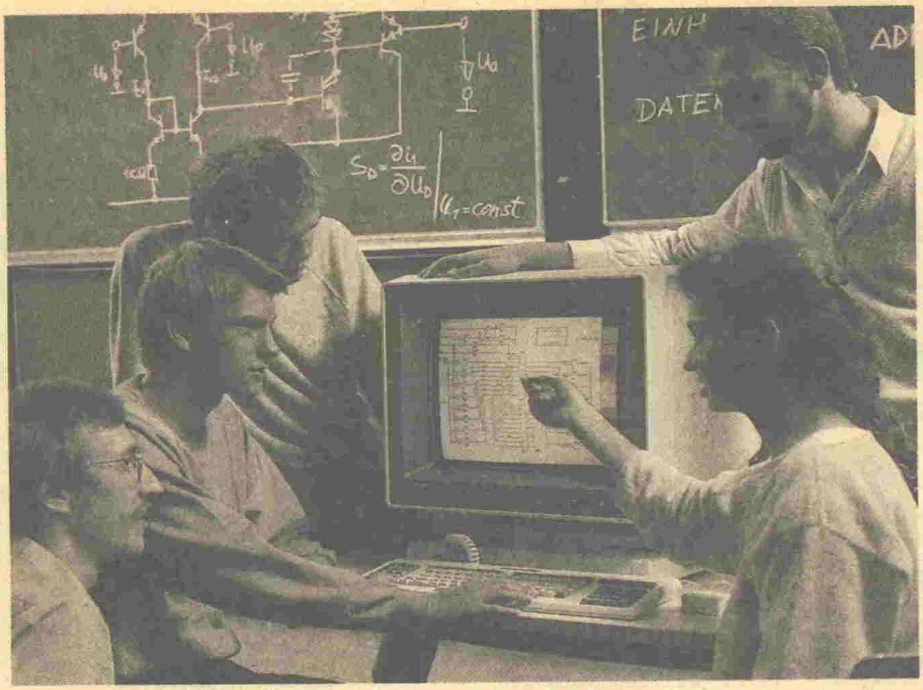
นั่นเป็นคำแนะนำที่ดีนะครับ ผมเองเคยพบเสมอ ๆ ว่านักวิเคราะห์และนักโปรแกรมมักจะไม่ค่อยสนใจในเรื่องการทดสอบมากนัก ส่วนใหญ่ก็มีเพียงข้อมูลและค่าผลลัพธ์ที่คิดว่าควรจะเป็นพอทดสอบแล้วพบว่าโปรแกรมให้ผลลัพธ์ตรงกับที่คาดเอาไว้ก็พอแล้ว ไม่ได้มีการคิดข้อมูลทดสอบให้รอบคอบครบถ้วนจนสามารถทดสอบคำสั่งทุกคำสั่งภายในโปรแกรมได้ ส่วนผลการทดสอบก็ไม่ค่อยเก็บเอาไว้เป็นหลักฐาน ผมคิดว่าถ้าต้องการจะพัฒนาซอฟต์แวร์อย่างมืออาชีพจริง ๆ ก็ควรจะต้องทำตามที่ ดร.เวอร์เนอร์ แซมเมอร์ เล่ามานี้แหละครับ

"ผมอยากทราบว่าตอนนี้ที่ซีเมนส์ใช้ภาษาคอมพิวเตอร์อะไรบ้างครับ"

"เราก็ใช้กันหลายภาษาครับ เช่น ภาษาเอด้า ภาษาซี ภาษาชิลล์ เป็นต้น" ดร.แซมเมอร์ตอบพร้อมกับขยายความต่อไปว่า "เราใช้ภาษาซี กับเครื่องที่ใช้ระบบปฏิบัติการ UNIX ตามที่ทราบ ๆ กันอยู่ แต่เวลานี้เรากำลังสนใจพัฒนาโปรแกรมภาษา Ada มากเพราะอย่างที่ทราบนั่นแหละครับว่า กระทรวงกลาโหมสหรัฐอเมริกาเขากำหนดให้ใช้ภาษานี้ ถ้าเราจะขายสินค้าให้เขาก็ต้องใช้ภาษานี้ มิฉะนั้นก็ขายให้เขาไม่ได้ ส่วนภาษา CHILL นั้นเป็นภาษาสำหรับงานโทรคมนาคมซึ่งกำหนดโดยคณะกรรมการ CCITT ซึ่งเราก็ใช้เหมือนกัน อ้อ...ผมเกือบลืมไป เราใช้ภาษาปาสคาลเหมือนกัน แต่เป็น Extended Pascal ซึ่งมีอะไร ๆ พิเศษต่างจากปาสคาลธรรมดา ภาษานี้เราใช้สำหรับงานประจำภายในของบริษัทซีเมนส์เองครับ"

"ผมอยากกลับมาถามเรื่อง Quality Assurance ของซอฟต์แวร์สักหน่อย ไม่ทราบว่าคุณมีเทคนิควิธีอย่างไรบ้างครับ"

"โอ...เรื่องคุณภาพนี้เราเน้นมากเลยครับ เรื่องนี้เราเริ่มครบทั้งวงจรของระบบเลยทีเดียว เริ่มแรกสุดเราพิจารณาคุณภาพของ Requirement เราตรวจสอบว่าการเขียน Requirement นั้นครบถ้วนสมบูรณ์หรือไม่ มีที่ใดตอนใดที่ขัดแย้งกันหรือไม่ ข้อความที่เขียนก็ต้องรัดกุมมาก อย่างเช่นการคำนวณค่าของฟังก์ชันนั้น เราจะไม่กำหนดดอกครับว่า "ต้องคำนวณค่าฟังก์ชันให้ถูกต้อง" เพราะกำหนดแบบนี้มันทำไม่ได้ เราจะกำหนดว่า "ต้องคำนวณฟังก์ชันให้มีความคลาดเคลื่อนไม่เกิน 0.00001" อะไรทำนองนี้ หรือในเรื่องของ Portable นั้นมันก็ไม่มีความหมาย เราควรกำหนดให้ชัดเจนว่าจะใช้ภาษาอะไรอย่างนี้จะดีกว่า ถัดจากการตรวจ Requirement ก็เป็นการตรวจสอบคุณภาพของการออกแบบซึ่งเราจะต้องพิจารณาแยกเป็นสองส่วน ส่วนแรกคือพิจารณาในแง่ของฟังก์ชัน คือจะต้องตรวจดูว่าระบบสามารถสนองตอบความต้องการที่กำหนดใน Requirement ได้ครบถ้วนหรือไม่ สำหรับส่วนที่สองนั้นพิจารณาในแง่ของโครงการ คือดูว่างานออกแบบนั้นถ้าทำสำเร็จแล้วจะมีประสิทธิภาพดีหรือไม่ ต่อจากนั้นเมื่อถึงขั้นการเขียนโปรแกรมก็เป็นการตรวจสอบโปรแกรมว่าเขียนขึ้นตามข้อกำหนดและมาตรฐานหรือไม่ ผมอยากแนะนำว่าสิ่งที่สำคัญที่สุดของการบริหารโครงการซอฟต์แวร์ก็คือ มาตรฐานและการควบคุม คุณควรกำหนดมาตรฐานต่าง ๆ เอาไว้ให้พร้อมเพรียง ไม่ว่าจะเป็นมาตรฐานของการใช้คำสั่งในภาษาที่เลือกใช้ มาตรฐานการตั้งชื่อตัวแปร และชื่อแฟ้มข้อมูล พอคุณมีมาตรฐานแล้วการตรวจสอบ



คุณภาพก็เป็นเรื่องง่าย" ดร.แซมเมอร์อธิบายเสียยืดยาว

"ดร.แซมเมอร์ครับ ผมทราบดีว่าบริษัทซีเมนส์เป็นบริษัทที่มีกิจการกว้างขวางใหญ่โต ดังนั้นบริษัทก็คงจะมีหน่วยงานวิจัยและพัฒนาที่กว้างขวางใหญ่โตมาก จะช่วยกรุณาเล่าให้ฟังได้สักหน่อยไหมครับว่า ปัจจุบันบริษัทสนใจทำงานวิจัยด้านใดบ้าง" ผมซัก

"ผมคงตอบไม่ได้แน่ ๆ เชียว" ดร.แซมเมอร์หัวเราะ "งานวิจัยของซีเมนส์เรากว้างขวางอย่างที่คุณว่า เราค้นคว้าทั้งทางด้านคอมพิวเตอร์ สารสนเทศ ไมโครอิเล็กทรอนิกส์"

"ลองเล่าทางด้านคอมพิวเตอร์ก็แล้วกันครับ" ผมต่อรอง

"ในทางด้านคอมพิวเตอร์นั้น เรามีวิศวกรทำงานฝ่ายวิจัยกว่าเจ็ดร้อยคน ส่วนมากได้ปริญญาเอกทางคอมพิวเตอร์ งานที่เราวิจัยทางด้านฮาร์ดแวร์ก็มีงานเกี่ยวกับสถาปัตยกรรมคอมพิวเตอร์ สถาปัตยกรรมแบบ Risc หรือ Reduced instruction set computer สถาปัตยกรรมที่ไม่ได้เป็นแบบฟอน นอยมานน์ ระบบ CAD และ Silicon compiler ส่วนงานวิจัยด้านซอฟต์แวร์ก็มีทางด้านภาษาคอมพิวเตอร์ การสร้างคอมพายเลอร์ ปัญญาประดิษฐ์ และ Software Tools" ดร.แซมเมอร์ตอบ

"ผมอยากให้ช่วยเล่าเรื่องของ Software Tools เพิ่มอีกสักหน่อยสิครับ ถ้าเข้าใจไม่ผิดผมคิดว่ามีคนทำเครื่องมือซอฟต์แวร์ประเภท Syntax directed editor เสร็จไปเยอะแล้ว ทางซีเมนส์จะยังมีอะไรต้องคิดค้นอยู่อีกหรือครับ" ผมถามไปแล้วก็คิดถึงเรื่องของวิศวกรรมซอฟต์แวร์ที่ผมเคยสนทนากับ นายมาซาโอะ มัตสุโมโต แห่งบริษัทเอนอีซี ไปสองครั้งแล้ว ท่านผู้อ่านหลายท่านคงจำได้ว่าที่นั่นเขาสนใจเรื่อง Software Tools มากทีเดียว

"มีสิครับ ผมคิดว่าเรายังต้องคิดค้นอะไรเพิ่มเติมอีกมาก ผมคิดว่าเราอาจจะสามารถทำ editor ที่ตรวจสอบ semantic อย่างง่าย ๆ ได้ แต่สิ่งที่จะตรวจสอบคงเป็นเรื่อง static เท่านั้น เรื่องที่เป็น dynamic คือมีการเปลี่ยนแปลงไปหลังจากโปรแกรมเริ่มทำงานแล้วนั้น editor นี้คงจะตรวจไม่ได้ ส่วนอีกเรื่องหนึ่งก็คือเรื่อง User interface เราเชื่อว่างานในอนาคตคงจะซับซ้อนมากกว่านี้และมีการผสมผสานเทคโนโลยีหลายอย่างเช่น ทางด้านตัวอักษร ทางด้านภาพกราฟิก ทางด้านเสียง ทางด้าน animation เป็นต้น บรรดาผู้ใช้ที่ไม่ค่อยรู้เรื่องคอมพิวเตอร์หรือไม่มีเวลาจะเรียนรู้คอมพิวเตอร์ คงจะไม่สามารถสร้างระบบหรือใช้คอมพิวเตอร์ได้ ดังนั้นเราจึงต้องการจัดทำเครื่องมือสำหรับช่วยผู้ใช้ที่จุดนี้"

"ดีครับ" ผมพูด "พอได้ฟังคำอธิบายผมก็นึกออกว่ายังมีเรื่องที่น่าพัฒนาทางนี้อีกมาก แล้วทางด้านปัญญาประดิษฐ์ล่ะครับ ที่ซีเมนส์มีโครงการอะไรเกี่ยวข้องกับงานซอฟต์แวร์หรือเปล่า"

"ก็พอมีบ้างครับ" ดร.แซมเมอร์ตอบ "ส่วนมากเป็นงานวิจัยพื้นฐานทางทฤษฎี ส่วนในด้านประยุกต์นั้นเราวิจัยกับงานด้าน CAD ครับ คือเรากำลังสร้างฐานความรู้สำหรับเก็บกฎเกณฑ์ต่าง ๆ เกี่ยวกับการคำนวณค่าทางไฟฟ้า ซึ่งจะช่วยให้นักออกแบบทำงานได้รวดเร็วขึ้น"

ครับ...นั่นก็คือเนื้อหาสาระที่ผมได้จากการนั่งสนทนากับ ดร.เวอร์เนอร์ แซมเมอร์ ผู้อำนวยการอาวุโส ฝ่ายวิจัยของบริษัทซีเมนส์ ในช่วงพักดื่มกาแฟระหว่างการบรรยายวิชาการ แม้ในความเป็นจริงเราจะไม่ได้ยินได้ฟังเรื่องที่ใหม่เอี่ยมจริง ๆ แต่เรื่องที่เรานำมาสนทนากันนั้น เป็นพื้นฐานสำคัญของการพัฒนาซอฟต์แวร์และอุตสาหกรรมซอฟต์แวร์นะครับ อุตสาหกรรมซอฟต์แวร์ของไทยจะยังเกิดไม่ได้จนกว่าเราจะรู้จักพัฒนาซอฟต์แวร์อย่างเป็นระบบ รู้จักใช้เทคนิคการพัฒนาที่ก้าวหน้า รู้จักการจัดการโครงการอย่างมีประสิทธิภาพ และมีการตรวจสอบควบคุมคุณภาพอย่างรัดกุม

ถ้าท่านผู้อ่านสงสัยว่าท่านพร้อมจะก้าวไปสู่งานอุตสาหกรรมซอฟต์แวร์แล้วหรือยัง ก็ขอให้ลองพิจารณางานของตัวท่านเองเทียบกับงานของบริษัทซีเมนส์ดูก็แล้วกันครับ...